NON-LINEAR REGRESSION ANALYSIS: A GENERAL PROGRAM FOR DATA MODELING USING PERSONAL MICROCOMPUTERS

D. P. NELSON
L. D. HOMER

REPORT NO. 83-5

NAVAL HEALTH RESEARCH CENTER
P. O. BOX 85122
SAN DIEGO, CALIFORNIA 92138

NAVAL MEDICAL RESEARCH AND DEVELOPMENT COMMAND
BETHESDA, MARYLAND

NON-LINEAR REGRESSION ANALYSIS:

A GENERAL PROGRAM FOR DATA MODELING

USING PERSONAL MICROCOMPUTERS

DENNIS P. NELSON, Ph.D. *
LCDR, MSC, USN

NAVAL HEALTH RESEARCH CENTER
P. O. Box 85122
SAN DIEGO, CA 92138

and

LOUIS D. HOMER, M.D., Ph.D.

NAVAL MEDICAL RESEARCH INSTITUTE
BETHESDA, MD 20814

Report No. 83-5, supported by Naval Medical Research and Development Command, Bethesda, Maryland, Department of the Navy, under research Work Unit 3M162770A871.AB.306. The views presented in this paper are those of the authors. No endorsement by the Department of the Navy has been given or should be inferred.

* Biological Sciences Department

ACKNOWLEDGEMENT

We wish to thank Dr. William R. Griswold, Department of Pediatrics, UCSD Medical School for providing the experimental data set upon which the examples are based.

SUMMARY

This report documents a general non-linear regression program for fitting data to non-linear models. The program is based on an algorithm by Marquardt which uses a least squares criterion to calculate successive improvements to an initial set of parameter estimates. The program is written in the BASIC language common to most microcomputers, because it is easy to use and to transport between machines from different manufacturers. The majority of inexpensive microcomputers do not offer matrix operations as part of their BASIC interpreter. The program presented here, therefore, supplies subroutines in BASIC for the zeroing, transposing and inverting of the required matrices, to make it compatible with most microcomputers available today. The report gives examples and program output based on a demonstration data set involving antigen-antibody complexation in solution. Two derivations of function subroutines are given to assist the user in developing his own functions. A complete listing of the necessary programs is given along with a section on program cautions.

INTRODUCTION

The advent of the personal computer and the development of general purpose software has now made it possible for bio-medical scientists to take advantage of powerful modeling tools previously available only to mathematicians or computer scientists. Care must be taken, however, to insure that programs are correctly applied and that any limitations in the methods are clearly understood.

One modeling application which appears repeatedly in bio-medical research is that of predicting an outcome on the basis of experience. This statistical method, known as regression analysis, requires that a functional relationship between the dependent and independent variables be specified. In the past, regression analysis has been largely limited to linear models. Such models can be solved by hand with a single matrix inversion, although they are more easily solved using a computer. Certain other models can be made linear by parameter transformation in order to utilize linear regression techniques. Transformations can introduce unwanted and sometimes unsuspected limitations or assumptions into the model and must be used with care. The majority of models encountered in bio-medical research and in problems of solution equilibrium, however, are non-linear and many cannot be transformed into linear form. In order to adequately model these systems a requirement exists for a general purpose non-linear regression technique which is both sufficiently general and robust to be applicable to a wide variety of research interests.

This report documents a general non-linear regression program for fitting data to non-linear models. The program is written in the BASIC language common to most microcomputers. The program has been adapted for use on several, stand-alone, microcomputer systems including the Wang 2200 series, the Radio Shack TRS-80 Models I and II, and the Apple II. Because BASIC is common to most microcomputer systems the program is transportable between machines with relative ease. Some microcomputers offer matrix operations as part of their BASIC interpreter, however the majority of inexpensive microcomputers do not. The program presented here supplies subroutines in BASIC for the zeroing, transposing and inverting of the required matrices, to make it compatible with most microcomputers available today. If the program is to be used on equipment which supports matrix operations, the matrix subroutine calls can be replaced with the corresponding matrix keywords.

METHOD

The program is based on an algorithm by Marquardt (1) which uses a Taylor's series expansion to give successive improvements to an initial set of parameter estimates. The method is actually a compromise between the Taylor's series (linearization method) and the method of steepest descent (gradient method). It combines the best features of both methods while avoiding their most serious limitations (2). It shares with the gradient methods their ability to converge from an initial guess which may be outside the region of convergence of other methods, and with the Taylor series methods their ability to rapidly converge once in the vicinity of the minimum. An attenuation parameter (lambda) is used to interpolate between the two methods as needed. Making lambda large favors the gradient method, and expands the region of convergence; making lambda small selects the Taylor series and favors rapid convergence. Although no single method can be considered best for all non-linear problems Marquart's method is a sensible first choice, provided the initial parameter estimates are reasonable.

The user supplies a subroutine which specifies the calculation of an error between an observed variable and a model of the observations. The model is a mathematical relationship between the independent variables (X(J,1)- X(J,5)); the coefficients or parameters being estimated (B(1)-B(10)); and the independent variable (Y(J)). The subscript (J) is used by the program to indicate each distinct data point and must be included in the model. The user provides the data and the starting guesses for the parameters or coefficients (B's) being estimated by the least squares criterion. The program then calculates a sum of squared errors (SSE). The program also calculates numerical partial derivatives of the function with respect to the B's and uses these together with the attenuation parameter (lambda) to select a new set of B's. If the new B's give a better SSE then lambda is divided by 10 and the cycle is repeated. If the new B's do not improve the SSE, then lambda is multiplied by 10 and another set of B's is calculated. At each iteration the new parameters, the SSE, lambda, and R-squared are printed out. All this is repeated until the pre-selected number of cycles has been exhausted or until successive changes in the B's are less than one part in 10,000 (Line 470). Detailed accounts of this method may be obtained from the references listed.

THE PROGRAM

The program actually exists in four separate parts (listed in Appendix A) which are stored as independent disk files. As listed they run on a Radio Shack TRS-80 Model II but can easily be adapted to any machine using Microsoft BASIC. The program statements that differ from machine to machine are those concerned with I/O. On some machines the multiple statement lines will need to be listed as single statements. Although several versions of this program have contained subroutines for plotting the data points as well as the functional relationship in order to visualize the model "fit", plotting routines have been omitted from this report because they are extremely machine dependent. If plotting is desired it is usually quite simple to add plot routines for a specific machine. The experimental data points are contained in the variable arrays Y() and X(,). A series of X values can be generated incrementally between a lower and an upper limit, and a corresponding Y value can be calculated for each using the function subroutine in Line 50 of the program. The model curve can then be plotted and compared to the plot of the experimental data. Of course only one independent variable can be plotted at a time.

The first program is called "NONLIN/DAT" and is used to facilitate data entry and to create disk data files. Many independent data sets can be created and stored on disk under different names using "NONLIN/DAT". The main program is called "NONLIN/REG" and performs the actual regression analysis. The third is "NONLIN/PRT" and cannot be used independently but may be merged with "NONLIN/REG" to divert the output from the screen to the printer. The fourth is the function subroutine. It is called by any 8 character description "********" plus the file extension /FUN. The function subroutine is a BASIC merge file and must be merged with "NONLIN/REG" prior to program execution. This design allows many functions to be stored on the disk with only a single copy of the "NONLIN/REG" program. Since both "NONLIN/PRT" and "********/FUN" are merge files they must be saved to disk in ASCII format.

Program 1 "NONLIN/DAT" and program 3 "NONLIN/PRT" are self explanatory, require no changes and should be used as listed except for any machine specific modifications. Program 2 "NONLIN/REG" has the option of adding a "Plot Data Points" routine between lines 1006 and 1499 and a "Plot Least Squares Fit" between lines 1506 and 1994. A subroutine to scale the data or to establish

upper and lower limits for the plot can be added in this same region. It is possible to change the convergence criterion by changing the value in line 470 to a larger or smaller number. The rest of the program is quite general with the ability to handle up to 100 data points and up to 5 independent variables (X(J,1)-X(J,5)) automatically. The total number of data points and the number of X variables are read in with the data set. The number of parameters is governed by the function and is limited to 10 (B(1)-B(10)). If computer RAM storage is limited it may be necessary to reduce the size of the X(,) and Y() arrays to conserve space. The program itself occupies about 5400 bytes of RAM and increases to 6000 bytes when the printer routine and an average function routine are added. Program space can be reduced to about 4400 bytes by compression if absolutely necessary. This is not recommended, however, because the program code becomes unreadable and the merging of printer and function routines impossible. The data arrays require an additional 5500 bytes for a total program requirement of 11,500 bytes. The best way to recover space is to redefine the data arrays, if possible, to use less memory. The majority of microcomputers can, however, accommodate this program easily in its entirety. In general programs 1-3 require little operator intervention or attention.

Program 4 "*********/FUN" is the only part of the program that is really variable and must be written by the operator. This is the mathematical function or model to which the data are to be fitted. The subroutine begins in line 50 and can continue through line 99. It is located in the beginning of the program for easy visibility and because it is the most frequently called routine in the program. Since the BASIC interpreter starts searching for subroutines at line 1, the location of the function subroutine early in the program provides a slight speed advantage. The subroutine is created like any other BASIC program and then saved to disk in ASCII format so that it can be merged into the main program. Since BASIC does not differentiate between variables in the main program and those in subroutines, care must be taken not to use any variables in the subroutine which are used in the main program. A variable cross reference index is given in Appendix D to help users avoid main program variables in the function subroutine. Array, string and explicitly defined variables of the same name, however, are all distinct in Microsoft BASIC. Appendix C gives the derivation of the function subroutines which correspond to those used for the example program output in Appendix B. The function routine has only two tasks, to use the parameters (B's) and the X values to calculate a Y(calc) value and to compare

this value with the true Y value to determine the error. This error must be stored in the variable E (E=Y(J)-Y(calc)) and will be used by the main program to compute the SSE. The subroutine must then return to the main program. It is a good practice to print the function subroutine (LLIST 50-99) before running any program. This provides a permanent record of the model used for that fit on the same page as the program output.

RUNNING THE PROGRAM

Most microcomputers first load a Disk Operating System (DOS). To run a BASIC program the BASIC interpreter must be loaded before the program. Allowance for at least one I/O buffer must be made when BASIC is loaded. The sequence for loading NONLIN is: LOAD "NONLIN/REG"; MERGE "NONLIN/PRT" if the output is to go to the printer; and MERGE "********/FUN" (the function subroutine which has been previously created and stored on disk). Examples of program output for one, two and three parameter models using the same data are given in Appendix B. Typing RUN begins execution of the program. The program then requests the following inputs:

Data File Name? --- Response: Enter the name given the data file at the time of file creation.

Max. No. Iterations? --- Response: Enter a limiting number of iterations in case the model does not converge, a good practical number is 15.

Initial Value Lambda? --- Response: Enter the starting value for lambda, usually 1. The more accurately the starting B's are known the smaller lambda can be.

No. of Parameters? --- Response: Enter number of parameters in the regression model. Must be the same as the # of parameters used in the function B(1) to B(10).

Starting Parameters? --- Response: Enter the best estimate possible for each of the parameters used in the function. If, for example, 3 was entered under No. of Parameters then a separate estimate must be given for each of the 3 parameters here. It is possible to defeat the program by entering unrealistic values for the B's at this point. The estimates must correspond exactly as used in the model B(1), B(2), B(3) etc.

Print Data? --- Response: Typing 1 will cause the data set to be printed out, Y first then each of the five X's. If any of the X's are not used a 0 is printed instead.

From this point on the program runs automatically printing the number for each iteration, the values of the parameters, the SSE, the R-square and lambda. Only those steps in which the SSE improves count as an iteration. If the SSE has increased instead of decreased, a new set of B's will be found using the old B's

and a 10 fold larger value for lambda. Lambda is not printed and the iteration counter is not incremented. Iteration continues until either the convergence criterion is met or the program runs out of iterations as specified on program entry. It then sets lambda=0 and calculates a final set of parameters, a final SSE and a final R-square. It also gives the overall variance and the standard deviation and an approximate standard error for each parameter with its coefficient of variation. The more nearly linear the model, the more accurate are these approximations.

The program then offers the option to print out the variance-covariance matrix, the correlation matrix and a table of residuals. The table of residuals is useful for quickly detecting bias in the model, and the correlation matrix for spotting interactions between parameters. The program also offers a complete printout of the X and Y data and uses the best values of the parameters (B's), found by the iteration, to compute the corresponding Y(calc) values and their approximate standard errors. The 95% confidence interval on Y(calc) is also printed as (+/- 2 SD) as is the residual. Only the first value of X is printed. If plotting subroutines have been added to the program, plots of the data and function can be drawn at this time. The program then either processes another data set or terminates.

EXAMPLES

Three separate program outputs for a one, a two and a three parameter model are provided in Appendix B. The same data set is used for all three examples to demonstrate how changing the mathematical model to more accurately represent the actual species in solution results in a considerable improvement in the fit of the experimental data. Derivation of the functions used for these models is provided in Appendix C to illustrate two common methods for developing the model functions. The data set used for the examples represents a number of serial dilutions of a solution containing an antigen-antibody complex. The Y value represents that fraction of the total antigen bound to antibody. The Y(J)'s were obtained by dividing the amount of radioactivity present in an ammonium sulfate precipitation of the complex by the total radioactivity. The X(J,1) value is the dilution factor for each of the 11 solutions. The total starting concentrations for both antigen and antibody are known. The total concentrations for each dilution are obtained by multiplying the total concentrations by the

dilution factor X(J,1). You will note that the undiluted solution is 33.4% bound while the 1/50 dilution is only 9.3% bound.

The first example in Appendix B is the 1:1 model for antigen-antibody binding. Only one parameter is needed for this model namely the equilibrium binding constant K. This model is the simplest case for a solution interaction since it permits only a single species of both antigen and antibody and only one interaction between them. Models of this type are relatively easily solved algebraically in terms of known quantities (such as the total antigen and total antibody concentrations) and the parameters being estimated (see Appendix C). The equilibrium expression and the mass balance equations reduce to a quadratic equation in either free antigen or free antibody concentration which can be solved using the quadratic equation as is done here. This method is preferable to iterative methods for function solution since the evaluation is direct, not by successive approximation. There are many instances, particularly with multiple equilibria, where higher order equations are encountered which are more difficult to solve algebraically. In these cases numerical procedures for evaluating the function are preferred although computer time may be lengthened substantially. The numerical solution of a function is shown in example 3. Many problems in solution equilibrium can be expressed in terms of the fraction of some component bound as a function of an unbound species. This type equation often reduces to the form (Y=B*X/(1+B*X) where Y is the fraction of a species bound, B is a parameter, usually a binding constant, and X is the free concentration of the binding species. The single parameter model yields a binding constant of 5.1E+5, with a standard error of about 17%. The SSE is .0323 and the R-square of .43 shows that only 43% of the total variation is explained by the model. The table of residuals shows a clear bias in the fit with large negative errors at one end of the data to large positive errors at the other. This is a clear indication that the model is not quite correct.

The second example is an attempt to improve the fit by introducing a second parameter but using the same 1:1 binding model. Here we postulate that the starting total antibody concentration is incorrect and that some other "effective" antibody concentration more accurately represents conditions in the solution. The first parameter is the binding constant and the second is the "effective" total antibody concentration. As you can see addition of this parameter reduces the SSE to .0016 and improves the R-square to 97% explained variation, a clear improvement in the fit. The binding constant is changed to

2.7E+6 with a standard error of 12%. The total antibody concentration drops from 4.5E-6 to an effective concentration of 2.3E-6 which greatly improves the fit to the 1:1 model. The standard error in this concentration parameter is only 4%. The residuals, although smaller than those in example 1, still indicate a bias since they are first positive, then negative and then positive again. In general one hopes for randomly distributed signs on the residuals. The correlation matrix indicates a fairly high correlation between the two variables at .84. On statistical grounds, an increase in the number of degrees of freedom (parameters) in the model is expected to decrease the SSE and improve R-square. Whether or not the decrease in the SSE, on addition of a parameter, is significant can be determined using an F-test.

Example 3 expands the model from a 1:1 single species binding model to a multiple species model. This 3 parameter model postulates that the total antibody concentration is divided between two different antibody species, each with a different binding constant, competing for a single species of free antigen. The inverse model where two different species of antigen compete for free antibody also fits the data well. The first two parameters are the two binding constants and the third is a distribution coefficient which determines what fraction of the total antibody belongs to each species. These equilibrium expressions and the mass balance equations would reduce algebraically to a cubic equation in the known quantities and the parameters. In general the resultant equation is a polynomial of order (# binding constants + 1). Solving higher order equations in terms of knowns and finding their roots, is more involved than the solution of a quadratic in examples 1 and 2. Since equilibrium equations are positive definite functions, an easier method is available, although it requires more computer time. It is possible to use a binary search to find the true value for free antibody concentration by successive approximation.

The binary search works as follows. First one uses the equilibrium and mass balance equations to solve for the free concentrations of antibody 1 (B1) and antibody 2 (B2) in terms of the binding constants (K1) and (K2), the total antibody concentration (TB) and the free antigen concentration (G). We also solve for total antigen concentration (TG) in terms of the (K1, K2, B1, B2) and (G). All except (G) are known. We can find (G) by guessing a concentration half way between zero and the total antigen concentration (TG). We then use this guess to calculate a value for TG, TG(calc). We compare this with the true value for TG. If TG(calc) is too high then the guess for G was too high and we set the

new upper limit on G to the old value for G and keep the lower limit. If TG is too low then G was too low and we set the lower limit on G to the old G value and keep the upper limit. A new G is then chosen half way between the new upper and lower limits and the procedure is repeated. The search continues until the new value for G does not differ significantly from the old value. The function then uses (TG-G)/TG to calculate the fraction antigen bound, Y(calc), computes the error E and returns.

The results yield a high affinity antibody with a constant of 4.4E+6, and a low affinity species with a binding constant of 6.0E+4. These constants have coefficients of variation of 12% and 20% respectively. The third parameter tells us that the proportion of high affinity antibody is 39% and low affinity antibody 61%. You will also note that the SSE has been decreased to .00033 and the R-square increased to 99.4% explained variation. This is a significant improvement in fit over the two parameter model in example 2. The residuals also look very good. They are all small, have randomly distributed signs and show no particular bias. The correlation matrix shows a correlation of .45 between the two binding constants, .84 between K1 and the distribution constant (R) and .82 between K2 and R. On the basis of the random residuals, the low SSE and the high R-square we would conclude that the model in example 3 adequately represents the data. Although there may be other models which also fit this data, 2 antibodies 1 antigen is a good model.

PROGRAM CAUTIONS

The Taylor's series used in the program has a region of convergence. Making lambda large increases the size of this region, however it is still possible to give initial parameter estimates which are outside this region. When this happens the program may diverge instead of converging and the program will terminate with an error condition, either a division by zero or an overflow/underflow error. This can also happen if the model is incorrectly postulated or an algebraic error is made in the function derivation. It can also happen if there is an error in the data. If a program error happens, the data should be carefully checked for accuracy, the functional model checked for algebra and the starting parameters checked. If this doesn't help a new parameter set should be tried or a larger starting lambda value.

Many functions of interest to the researcher are limited in their range by nature. For example concentrations and therefore binding constants can only have positive values (negative concentrations are undefined). The regression program has no such limitation, however, since mathematically negative and positive numbers are equally valid. This can present a problem since the program can find solutions with negative (meaningless) as well as positive binding constants. The problem can be avoided by careful restructuring of the function i.e. by transforming one or more parameters to new parameters with natural limits corresponding to the desired limits. For example one can use the log K value instead of the K parameter since the log function is naturally limited to positive numbers. In example 3 we used another technique to limit the range of a parameter. Instead of using the parameter R for the distribution coefficient between the antibody species, we used the squared Sin function. Since the only meaningful values for R are positive numbers between 0 and 1 the range of R must be restricted. This is done by letting (R=Sin(B(3))*Sin(B(3)) where the new parameter is the argument of the Sin function in radians. Now R is limited to the positive range 0 to 1 whereas if it were a parameter itself it could include all positive and negative real numbers. The immediate mode of BASIC can be used after completion of the program to print the actual value of the parameter of interest.

One must also recognize that just because the program finds a parameter fit to a data set does not make that model true. The mathematical representation may be useful for prediction but may or may not shed any light on the actual mechanism of the process under investigation. The model should not only be algebraically correct it should also have either physical or chemical significance, i.e. it should represent and be consistent with all other known characteristics of the real system.

Certain models and data sets have an SSE surface which exhibits regional as well as global mimima. If this occurs it is possible for the program to locate a local minimum instead of the global minimum. In this case the convergence criterion is met but the program has fallen into a pit and has not located the lowest point on the surface. To test for this condition one should, when initially investigating a new model, examine the surface using several starting values for the parameters. If the program converges to the same point from several different starting points one can be reasonably confident that the global minimum has been found.

REFERENCES

1. Marquardt, D.W., "An Algorithm for Least-squares Estimation of Non-linear Parameters.", J. Soc. Indust. Appl. Math., 2, 431-441 (1963).

2. Draper, N.R. and Smith, H., Applied Regression Analysis, John Wiley & Sons, Inc., New York, N. Y. (1966).

PROGRAM (1) NONLIN/DAT

```
10 REM **** DATA INPUT ROUTINE FOR NONLIN ****
15 DIM Y(100),X(100,5)
20 CLS : INPUT "NUMBER OF DATA POINTS (100 MAX)";N
25 INPUT "NUMBER OF INDEPENDENT (X) VARIABLES (5 MAX)";N1
30 INPUT "NAME OF DATA FILE";Z$
35 PRINT : PRINT "Y VAR. , X VARS."
40 PRINT "----------------" : PRINT
45 FOR I=1 TO N
50 PRINT "ENTER Y(";I;")"; : INPUT Y(I)
55 FOR J=1 TO N1
60 PRINT "X(";I;J;")"; : INPUT X(I,J)
65 NEXT J,I
70 OPEN "O",1,Z$
75 PRINT #1, N,N1
80 FOR I=1 TO N
85 PRINT #1, Y(I)
90 FOR J=1 TO N1
95 PRINT #1, X(I,J)
100 NEXT J,I
105 CLOSE 1
110 END
```

PROGRAM (2) NONLIN/REG

```
10 CLS : PRINT "***** NON-LINEAR REGRESSION ANALYSIS ****" : PRINT
15 DEFINT I,J,K
20 DIM B(10),G(10),P(10),Y(100),X(100,5)
25 DIM A(10,10),AA(10,10),C(10,10),Q(10,10)
30 GOSUB 2400
35 GOTO 110
50 REM *** INSERT FUNCTION HERE (LINES 50-99) ***
*** THE FUNCTION MUST CALCULATE A VALUE (F) FOR Y(J),
*** COMPUTE THE ERROR BETWEEN THE OBSERVED AND CALCULATED
*** Y VALUES (E=Y(J)-F), THEN RETURN.
100 INPUT "NEW POINT SET (0=NO, 1=YES)";E6 : PRINT
105 IF E6>0 THEN RUN ELSE 1995
110 INPUT "MAX. NO. OF ITERATIONS";T1
120 INPUT "INITIAL VALUE OF LAMBDA";L
130 INPUT "NO. OF PARAMETERS";N1
140 PRINT : PRINT "STARTING PARAMETERS" : PRINT
145 FOR I=1 TO N1
150 PRINT "B(";I;")"; : INPUT B(I) : NEXT I
160 PRINT : INPUT "DO YOU WANT TO PRINT DATA (0=NO, 1=YES)";E5
170 IF E5=0 THEN 200
175 PRINT : PRINT "OBSERVATIONS  Y,X(1-5)" : PRINT
180 FOR I=1 TO N
190 PRINT "(";I;")";TAB(7);Y(I);TAB(17);X(I,1);TAB(27);X(I,2);
TAB(37);X(I,3);TAB(47);X(I,4);TAB(57);X(I,5)
195 NEXT I
200 Z1=0 : Z2=0
205 FOR I=1 TO N
210 Z1=Z1+Y(I)
215 Z2=Z2+Y(I)*Y(I)
220 NEXT I
225 T2=0
230 T2=T2+1
235 S0=0
240 GOSUB 2000
245 GOSUB 2025
250 GOSUB 2050
255 GOSUB 2075
260 FOR J=1 TO N
265 GOSUB 50
270 S0=S0+E*E
275 E1=E
280 FOR I=1 TO N1
285 B(I)=B(I)*1.001
290 GOSUB 50
295 B(I)=B(I)/1.001
300 P(I)=(E1-E)/(.001*B(I))
305 NEXT I
310 FOR I=1 TO N1
315 G(I)=G(I)+E1*P(I)
320 FOR I1=1 TO N1
325 A(I,I1)=A(I,I1) + P(I)*P(I1)
330 NEXT I1
335 NEXT I
340 NEXT J
345 FOR I=1 TO N1
350 FOR J=1 TO N1
355 Q(I,J)=A(I,J)/(SQR((A(I,I))*(A(J,J))))
360 NEXT J
365 G(I)=G(I)/(SQR(A(I,I)))
370 NEXT I
375 PRINT : PRINT "SSE=";S0;TAB(22);"ITERATION NO.=";T2;
TAB(49);"LAMBDA=";L
390 PRINT "R-SQUARE =";1-(S0/(Z2-(Z1*Z1/N)))
395 FOR I=1 TO N1
400 Q(I,I)=Q(I,I)*(1+L)
405 NEXT I
```

```
410 GOSUB 2250
415 FOR I=1 TO N1
420 P(I)=0
425 FOR J=1 TO N1
430 P(I)=P(I)+C(I,J)*G(J)
435 NEXT J
440 P(I)=P(I)/(SQR(A(I,I)))
445 NEXT I
450 IF T1<0 THEN 650
455 IF T2>=T1 THEN 500
460 NN=0 : I=1
465 IF I>N1 THEN 510
470 IF ABS(P(I)/B(I))<.0001 THEN NN=NN+1
475 IF NN>=N1 THEN 490
480 I=I+1 : GOTO 465
490 PRINT : PRINT "CONVERGENCE"
495 GOTO 620
500 PRINT "YOU ARE OUT OF ITERATIONS"
505 GOTO 620
510 FOR I=1 TO N1
515 B(I)=B(I)+P(I)
520 NEXT I
525 S1=0
530 FOR J=1 TO N
535 GOSUB 50
540 S1=S1+E*E
545 NEXT J
550 PRINT "PARAMETERS"
555 FOR I=1 TO N1
560 PRINT B(I);
570 NEXT I
575 PRINT
580 IF S1>S0 THEN 595
585 L=L/10
590 GOTO 230
595 L=L*10
600 FOR I=1 TO N1
605 B(I)=B(I)-P(I)
610 NEXT I
615 GOTO 395
620 T1=-1
625 L=0
630 GOTO 230
650 PRINT
655 V=S0/(N-N1)
660 V1=SQR(V)
665 PRINT "VAR=";V;TAB(22);"SD=";V1;TAB(49);"SSE=";S0
675 PRINT
680 GOSUB 2275
685 GOSUB 2300
690 PRINT "FINAL PARAMETERS";TAB(22);"STD. ERROR OF PARAM.";
TAB(49);"COEFF. OF VAR."
695 PRINT "----------------";TAB(22);"--------------------";
TAB(49);"--------------"
715 FOR I=1 TO N1
720 D=SQR(A(I,I))
725 PRINT TAB(3);B(I);TAB(27);D;TAB(52);D/B(I)
730 NEXT I
735 PRINT : PRINT "PRINT VARIANCE-COVARIANCE MATRIX
(0=NO,1=YES)";
740 INPUT E7 : PRINT
745 IF E7=0 THEN 765
750 GOSUB 2325
760 PRINT
765 PRINT "DO YOU WISH PRINTOUT OF ESTIMATED Y FOR EACH X
(0=NO,1=YES)";
770 INPUT H7 : PRINT
775 IF H7=0 THEN 920
```

```
780 T=2
785 PRINT TAB(6);"X";TAB(16);"Y";TAB(23);"ESTIMATED Y";TAB(36);
"S.E. EST. Y";TAB(49);"RESIDUAL"
790 PRINT TAB(23);"-(";T;"* S.E.)";TAB(36);"+(";T;"* S.E.)"
792 PRINT TAB(2);"--------";TAB(12);"---------";TAB(23);
"------------";TAB(36);"------------";TAB(49);"--------"
800 FOR J=1 TO N
805 GOSUB 50
810 E1=E
815 FOR I=1 TO N1
820 B(I)=B(I)*1.001
825 GOSUB 50
830 B(I)=B(I)/1.001
835 P(I)=(E1-E)/(.001*B(I))
840 NEXT I
845 V1=0
850 FOR J2=1 TO N1
855 FOR J1=1 TO N1
860 V1=V1 + A(J1,J2)*P(J1)*P(J2)
865 NEXT J1
870 NEXT J2
875 V2=SQR(V1)
880 L1=Y(J)-E1-T*V2
885 L2=L1+2*T*V2
890 PRINT TAB(2);X(J,1);TAB(12);Y(J);TAB(23);Y(J)-E1;TAB(36);
V2;TAB(49);E1
900 PRINT TAB(23);L1;TAB(36);L2
910 NEXT J
915 PRINT
920 PRINT "DO YOU WISH TABLE OF RESIDUALS (0=NO,1=YES)";
925 INPUT H8
930 IF H8=0 THEN 970
935 PRINT
940 PRINT "             RESIDUALS"
945 PRINT
950 FOR J=1 TO N
955 GOSUB 50
960 PRINT "(";J;")",E
965 NEXT J
970 PRINT : PRINT "PRINT CORRELATION MATRIX (0=NO,1=YES)";
975 INPUT E8 : IF E8=0 THEN 1000
980 PRINT
985 GOSUB 2350
1000 PRINT : INPUT "PLOT DATA POINTS (0=NO,1=YES)";D1 : PRINT
1005 IF D1=0 THEN 1500
1500 INPUT "PLOT LEAST SQUARES FIT (0=NO,1=YES)";D2 : PRINT
1505 IF D2=0 THEN 100
1995 END
2000 REM  ** ZERO MATRIX G **
2005 FOR II=1 TO N1
2010 G(II)=0
2015 NEXT II
2020 RETURN
2025 REM  ** ZERO MATRIX A **
2030 FOR II=1 TO N1 : FOR JJ=1 TO N1
2035 A(II,JJ)=0
2040 NEXT JJ : NEXT II
2045 RETURN
2050 REM  ** ZERO MATRIX Q **
2055 FOR II=1 TO N1 : FOR JJ=1 TO N1
2060 Q(II,JJ)=0
2065 NEXT JJ : NEXT II
2070 RETURN
2075 REM  ** ZERO MATRIX C **
2080 FOR II=1 TO N1 : FOR JJ=1 TO N1
2085 C(II,JJ)=0
2090 NEXT JJ : NEXT II
2095 RETURN
```

```
2100 REM * MATRIX INVERSION ROUTINE *
2105 FOR JJ=1 TO N1
2110 FOR II=JJ TO N1
2115 IF AA(II,JJ)<>0 THEN 2135
2120 NEXT II
2125 PRINT "SINGULAR MATRIX"
2130 STOP
2135 FOR KK=1 TO N1
2140 S=AA(JJ,KK)
2145 AA(JJ,KK)=AA(II,KK)
2150 AA(II,KK)=S
2155 S=C(JJ,KK)
2160 C(JJ,KK)=C(II,KK)
2165 C(II,KK)=S
2170 NEXT KK
2175 TT=1/AA(JJ,JJ)
2180 FOR KK=1 TO N1
2185 AA(JJ,KK)=TT*AA(JJ,KK)
2190 C(JJ,KK)=TT*C(JJ,KK)
2195 NEXT KK
2200 FOR LL=1 TO N1
2205 IF LL=JJ THEN 2235
2210 TT=-AA(LL,JJ)
2215 FOR KK=1 TO N1
2220 AA(LL,KK)=AA(LL,KK)+TT*AA(JJ,KK)
2225 C(LL,KK)=C(LL,KK)+TT*C(JJ,KK)
2230 NEXT KK
2235 NEXT LL
2240 NEXT JJ
2245 RETURN
2250 REM * TRANSFER MAT Q TO MAT AA *
2255 GOSUB 2075
2260 FOR II=1 TO N1 : FOR JJ=1 TO N1 : AA(II,JJ)=Q(II,JJ)
2265 NEXT JJ : C(II,II)=1 : NEXT II
2270 GOSUB 2100 : RETURN
2275 REM * TRANSFER MAT A TO MAT AA *
2280 GOSUB 2075
2285 FOR II=1 TO N1 : FOR JJ=1 TO N1 : AA(II,JJ)=A(II,JJ)
2290 NEXT JJ : C(II,II)=1 : NEXT II
2295 GOSUB 2100 : RETURN
2300 REM : CONFIDENCE MATRIX TIMES VARIANCE *
2305 FOR II=1 TO N1 : FOR JJ=1 TO N1
2310 A(II,JJ)=C(II,JJ)*V
2315 NEXT JJ,II
2320 RETURN
2325 REM * MATPRINT A *
2330 FOR II=1 TO N1 : FOR JJ=1 TO N1
2335 E8=A(II,JJ) : PRINT E8;
2340 NEXT JJ : PRINT : NEXT II
2345 RETURN
2350 REM * MATPRINT Q *
2355 FOR II=1 TO N1 : FOR JJ=1 TO N1
2360 E8=Q(II,JJ) : PRINT E8;
2365 NEXT JJ : PRINT : NEXT II
2370 RETURN
2400 REM ** DATA INPUT ROUTINE **
2405 INPUT "ENTER DATA FILE NAME";Z$
2410 OPEN "I",1,Z$
2415 INPUT #1, N,N2
2420 FOR I=1 TO N
2425 INPUT #1, Y(I)
2430 FOR J=1 TO N2
2435 INPUT #1, X(I,J)
2440 NEXT J,I
2445 CLOSE 1
2450 RETURN
```

PROGRAM (3) NONLIN/PRT

```
5 LPRINT "NONLINEAR REGRESSION ANALYSIS" : LPRINT
31 LPRINT "DATA FILE NAME",,Z$
115 LPRINT "MAX. NO. OF ITERATIONS",T1
125 LPRINT "INITIAL VALUE OF LAMBDA",L
135 LPRINT "NO. OF PARAMETERS",N1
141 LPRINT : LPRINT "STARTING PARAMETERS" : LPRINT
150 PRINT "B(";I;")"; : INPUT B(I) : LPRINT "B(";I;")",B(I) : NEXT I
175 LPRINT : LPRINT : LPRINT "OBSERVATIONS Y,X(1-5)" : LPRINT
190 LPRINT "(";I;")";TAB(7);Y(I);TAB(17);X(I,1);TAB(27);X(I,2);
TAB(37);X(I,3);TAB(47);X(I,4);TAB(57);X(I,5)
375 LPRINT : LPRINT "SSE=";S0;TAB(22);"ITERATION NO.=";T2;
TAB(49);"LAMBDA=";L
390 LPRINT "R-SQUARE =";1-(S0/(Z2-(Z1*Z1/N)))
490 LPRINT : LPRINT "CONVERGENCE"
500 LPRINT "YOU ARE OUT OF ITERATIONS"
550 LPRINT "PARAMETERS"
560 LPRINT B(I);
575 LPRINT
650 LPRINT : LPRINT
665 LPRINT "VAR=";V;TAB(22);"SD=";V1;TAB(49);"SSE=";S0
690 LPRINT : LPRINT : LPRINT "FINAL PARAMETERS";TAB(22);
"STD. ERROR OF PARAM.";TAB(49);"COEFF. OF VAR."
695 LPRINT "----------";TAB(22);"----------";
TAB(49);"----------"
725 LPRINT TAB(3);B(I);TAB(27);D;TAB(52);D/B(I)
746 LPRINT : LPRINT : LPRINT "VARIANCE-COVARIANCE MATRIX" : LPRINT
776 LPRINT : LPRINT : LPRINT "PRINTOUT OF ESTIMATED Y FOR EACH X" :
LPRINT
785 LPRINT TAB(6);"X";TAB(16);"Y";TAB(23);"ESTIMATED Y";
TAB(36);"S.E. EST. Y";TAB(49);"RESIDUAL"
790 LPRINT TAB(23);"-(";T;"* S.E.)";TAB(36);"+(";T;"* S.E.)"
792 LPRINT TAB(2);"--------";TAB(12);"---------";TAB(23);
"----------";TAB(36);"----------";TAB(49);"--------"
890 LPRINT TAB(2);X(J,1);TAB(12);Y(J);TAB(23);Y(J)-E1;TAB(36);
V2;TAB(49);E1
900 LPRINT TAB(23);L1;TAB(36);L2
940 LPRINT : LPRINT : LPRINT "TABLE OF RESIDUALS" : LPRINT
960 LPRINT "(";J;")",E
981 LPRINT : LPRINT : LPRINT "CORRELATION MATRIX" : LPRINT
1995 SYSTEM "T" : END
2335 E8=A(II,JJ) : LPRINT E8;
2340 NEXT JJ : LPRINT : NEXT II
2360 E8=Q(II,JJ) : LPRINT E8;
2365 NEXT JJ : LPRINT : NEXT II
```

PROGRAM (4) AGAB/FUN (1 PARAMETER)

```
50 REM ** FUNCTION FOR 1:1 ANTIGEN ANTIBODY BINDING **
55 P1=B(1) : TG=X(J,1)*6.9E-6 : TB=X(J,1)*4.5E-6
60 A1=P1 : B1=1+P1*TG-P1*TB : C1=-TB
65 AB=(-B1+SQR(B1*B1-4*A1*C1))/(2*A1)
70 F=P1*AB/(1+P1*AB)
75 E=Y(J)-F
80 RETURN
```

PROGRAM (4) AGAB2/FUN (2 PARAMETERS)

```
50 REM ** FUNCTION FOR 1:1 ANTIGEN ANTIBODY BINDING **
55 P1=B(1) : TG=X(J,1)*6.9E-6 : TB=X(J,1)*B(2)
60 A1=P1 : B1=1+P1*TG-P1*TB : C1=-TB
65 AB=(-B1+SQR(B1*B1-4*A1*C1))/(2*A1)
70 F=P1*AB/(1+P1*AB)
75 E=Y(J)-F
80 RETURN
```

PROGRAM (4) AB2AG/FUN (3 PARAMETERS)

```
50 REM * FUNCTION FOR 2 DISTINCT SPECIES OF ANTIBODY, DIFFERENT K'S *
52 P1=B(1) : P2=B(2) : TB=X(J,1)*4.5E-6 : TG=X(J,1)*6.9E-6
54 BL=0 : LG=0 : TL=TG
56 R=SIN(B(3))*SIN(B(3))
58 G=.5*(BL+TL)
60 B1=R*TB/(1+P1*G)
62 B2=(1-R)*TB/(1+P2*G)
64 TC=G*(1+P1*B1+P2*B2)
66 IF TC>TG THEN TL=G ELSE BL=G
68 IF ABS((LG-G)/G)<1E-6 THEN 72
70 LG=G : GOTO 58
72 F=(TG-G)/TG : E=Y(J)-F : RETURN
```

EXAMPLE (1) - FIXED TOTAL ANTIGEN AND ANTIBODY CONCENTRATION - (1 PARAMETER)

```
50 REM ** FUNCTION FOR 1:1 ANTIGEN ANTIBODY BINDING **
55 P1=B(1) : TG=X(J,1)*6.9E-6 : TB=X(J,1)*4.5E-6
60 A1=P1 : B1=1+P1*TG-P1*TB : C1=-TB
65 AB=(-B1+SQR(B1*B1-4*A1*C1))/(2*A1)
70 F=P1*AB/(1+P1*AB)
75 E=Y(J)-F
80 RETURN
```

NONLINEAR REGRESSION ANALYSIS

DATA FILE NAME	AGAB201/DAT
MAX. NO. OF ITERATIONS	15
INITIAL VALUE OF LAMBDA	1
NO. OF PARAMETERS	1

STARTING PARAMETERS

B(1) 1E+06

OBSERVATIONS Y,X(1-5)

(1)	.336	1	0	0	0	0
(2)	.271	.5	0	0	0	0
(3)	.247	.25	0	0	0	0
(4)	.232	.2	0	0	0	0
(5)	.231	.166667	0	0	0	0
(6)	.21	.125	0	0	0	0
(7)	.196	.1	0	0	0	0
(8)	.147	.05	0	0	0	0
(9)	.122	.0333333	0	0	0	0
(10)	.108	.025	0	0	0	0
(11)	.093	.02	0	0	0	0

SSE= .0788425 ITERATION NO.= 1 LAMBDA= 1
R-SQUARE =-.391782
PARAMETERS
723492

SSE= .0440415 ITERATION NO.= 2 LAMBDA= .1
R-SQUARE = .22255
PARAMETERS
534537

SSE= .032425 ITERATION NO.= 3 LAMBDA= .01
R-SQUARE = .427612
PARAMETERS
517085

SSE= .0322849 ITERATION NO.= 4 LAMBDA= 1E-03
R-SQUARE = .430085
PARAMETERS
514760

SSE= .0322822 ITERATION NO.= 5 LAMBDA= 1E-04
R-SQUARE = .430132
PARAMETERS
514392

SSE= .0322822 ITERATION NO.= 6 LAMBDA= 1E-05
R-SQUARE = .430133

CONVERGENCE

SSE= .0322822 ITERATION NO.= 7 LAMBDA= 0
R-SQUARE = .430133

EXAMPLE (1) - FIXED TOTAL ANTIGEN AND ANTIBODY CONCENTRATION - (1 PARAMETER)

VAR= 3.22822E-03 SD= .0568174 SSE= .0322822

FINAL PARAMETERS	STD. ERROR OF PARAM.	COEFF. OF VAR.
514392	87748.9	.170588

VARIANCE-COVARIANCE MATRIX

7.69987E+09

PRINTOUT OF ESTIMATED Y FOR EACH X

X	Y	ESTIMATED Y -(2 * S.E.)	S.E. EST. Y +(2 * S.E.)	RESIDUAL
1	.336	.435135 .395837	.0196493 .474434	-.0991355
.5	.271	.349467 .305227	.0221201 .393707	-.0784667
.25	.247	.258758 .214783	.0219879 .302734	-.0117584
.2	.232	.23041 .187827	.0212914 .272993	1.58979E-03
.166667	.231	.208058 .167076	.0204907 .249039	.0229425
.125	.21	.174782 .137009	.0188867 .212556	.0352179
.1	.196	.151015 .11621	.0174022 .185819	.0449853
.05	.147	.0906243 .0661224	.012251 .115126	.0563757
.0333333	.122	.0649603 .0461854	9.38744E-03 .0837352	.0570397
.025	.108	.0506688 .0354755	7.59664E-03 .0658621	.0573312
.02	.093	.0415453 .0287859	6.37967E-03 .0543046	.0514547

TABLE OF RESIDUALS

(1)	-.0991355
(2)	-.0784667
(3)	-.0117584
(4)	1.58979E-03
(5)	.0229425
(6)	.0352179
(7)	.0449853
(8)	.0563757
(9)	.0570397
(10)	.0573312
(11)	.0514547

CORRELATION MATRIX

1

EXAMPLE (2) - VARIABLE TOTAL ANTIBODY CONCENTRATION - (2 PARAMETERS)

```
50 REM ** FUNCTION FOR 1:1 ANTIGEN ANTIBODY BINDING **
55 P1=B(1) : TG=X(J,1)*6.9E-6 : TB=X(J,1)*B(2)
60 A1=P1 : B1=1+P1*TG-P1*TB : C1=-TB
65 AB=(-B1+SQR(B1*B1-4*A1*C1))/(2*A1)
70 F=P1*AB/(1+P1*AB)
75 E=Y(J)-F
80 RETURN
```

NONLINEAR REGRESSION ANALYSIS

```
DATA FILE NAME                  AGAB201/DAT
MAX. NO. OF ITERATIONS           15
INITIAL VALUE OF LAMBDA          1
NO. OF PARAMETERS                2

STARTING PARAMETERS

B( 1 )          1E+06
B( 2 )          4.5E-06
```

OBSERVATIONS Y,X(1-5)

```
( 1 )   .336    1          0    0    0    0
( 2 )   .271    .5         0    0    0    0
( 3 )   .247    .25        0    0    0    0
( 4 )   .232    .2         0    0    0    0
( 5 )   .231    .166667    0    0    0    0
( 6 )   .21     .125       0    0    0    0
( 7 )   .196    .1         0    0    0    0
( 8 )   .147    .05        0    0    0    0
( 9 )   .122    .0333333   0    0    0    0
( 10 )  .108    .025       0    0    0    0
( 11 )  .093    .02        0    0    0    0

SSE= .0788425          ITERATION NO.= 1          LAMBDA= 1
R-SQUARE =-.391782
PARAMETERS
856896  3.8962E-06

SSE= .0271986          ITERATION NO.= 2          LAMBDA= .1
R-SQUARE = .519872
PARAMETERS
1.13332E+06  2.84243E-06

SSE= .0103209          ITERATION NO.= 3          LAMBDA= .01
R-SQUARE = .817809
PARAMETERS
1.93685E+06  2.22722E-06

SSE= 7.65994E-03       ITERATION NO.= 4          LAMBDA= 1E-03
R-SQUARE = .864782
PARAMETERS
2.53919E+06  2.29288E-06

SSE= 1.65261E-03       ITERATION NO.= 5          LAMBDA= 1E-04
R-SQUARE = .970827
PARAMETERS
2.65473E+06  2.29025E-06

SSE= 1.57879E-03       ITERATION NO.= 6          LAMBDA= 1E-05
R-SQUARE = .97213
PARAMETERS
2.65934E+06  2.28993E-06
```

EXAMPLE (2) - VARIABLE TOTAL ANTIBODY CONCENTRATION - (2 PARAMETERS)

SSE= 1.57867E-03 ITERATION NO.= 7 LAMBDA= 1E-06
R-SQUARE = .972132
PARAMETERS
2.66445E+06 2.28895E-06

SSE= 1.57865E-03 ITERATION NO.= 8 LAMBDA= 1E-07
R-SQUARE = .972133
PARAMETERS
2.67128E+06 2.2874E-06

SSE= 1.5786E-03 ITERATION NO.= 9 LAMBDA= 1E-08
R-SQUARE = .972134
PARAMETERS
2.66406E+06 2.28876E-06

SSE= 1.57857E-03 ITERATION NO.= 10 LAMBDA= 1E-09
R-SQUARE = .972134
PARAMETERS
2.65847E+06 2.29005E-06
PARAMETERS
2.65847E+06 2.29005E-06
PARAMETERS
2.65847E+06 2.29005E-06
PARAMETERS
2.65847E+06 2.29005E-06
PARAMETERS
2.65847E+06 2.29005E-06
PARAMETERS
2.65847E+06 2.29005E-06
PARAMETERS
2.6585E+06 2.29004E-06
PARAMETERS
2.65881E+06 2.28996E-06
PARAMETERS
2.66064E+06 2.28949E-06
PARAMETERS
2.66324E+06 2.28887E-06

CONVERGENCE

SSE= 1.57857E-03 ITERATION NO.= 11 LAMBDA= 0
R-SQUARE = .972134

VAR= 1.75397E-04 SD= .0132437 SSE= 1.57857E-03

FINAL PARAMETERS	STD. ERROR OF PARAM.	COEFF. OF VAR.
2.66406E+06	322641	.121109
2.28876E-06	8.48315E-08	.0370644

VARIANCE-COVARIANCE MATRIX

1.04097E+11 -.0229482
-.0229482 7.19637E-15

PRINTOUT OF ESTIMATED Y FOR EACH X

X	Y	ESTIMATED Y -(2 * S.E.)	S.E. EST. Y +(2 * S.E.)	RESIDUAL
1	.336	.307543 .289499	9.02226E-03 .325588	.0284566

EXAMPLE (2) - VARIABLE TOTAL ANTIBODY CONCENTRATION - (2 PARAMETERS)

.5	.271	.287748	6.86438E-03	-.0167481
		.274019		.301477
.25	.247	.256595	4.88633E-03	-9.59546E-03
		.246823		.266368
.2	.232	.243942	4.5436E-03	-.011942
		.234855		.253029
.166667	.231	.23271	4.44295E-03	-1.70967E-03
		.223824		.241596
.125	.21	.213538	4.57805E-03	-3.53788E-03
		.204382		.222694
.1	.196	.197674	4.84864E-03	-1.67352E-03
		.187976		.207371
.05	.147	.145878	5.58837E-03	1.12191E-03
		.134701		.157055
.0333333	.122	.1165	5.53572E-03	5.49953E-03
		.105429		.127572
.025	.108	.0972603	5.24935E-03	.0107398
		.0867616		.107759
.02	.093	.0835914	4.91065E-03	9.40858E-03
		.0737701		.0934127

TABLE OF RESIDUALS

(1)	.0284566
(2)	-.0167481
(3)	-9.59546E-03
(4)	-.011942
(5)	-1.70967E-03
(6)	-3.53788E-03
(7)	-1.67352E-03
(8)	1.12191E-03
(9)	5.49953E-03
(10)	.0107398
(11)	9.40858E-03

CORRELATION MATRIX

.999997	.838443
.838443	1

EXAMPLE (3) - 2 SPECIES ANTIBODY, 2 BINDING CONSTANTS - (3 PARAMETERS)

```
50 REM ** FUNCTION FOR 2 DISTINCT SPECIES OF ANTIBODY, DIFFERENT K'S **
52 P1=B(1) : P2=B(2) : TG=X(J,1)*6.9E-6 : TB=X(J,1)*4.5E-6
54 BL=0 : LG=0 : TL=TG
56 R=SIN(B(3))*SIN(B(3))
58 G=.5*(BL+TL)
60 B1=R*TB/(1+P1*G)
62 B2=(1-R)*TB/(1+P2*G)
64 TC=G*(1+P1*B1+P2*B2)
66 IF TC>TG THEN TL=G ELSE BL=G
68 IF ABS((LG-G)/G)<1E-6 THEN 72
70 LG=G : GOTO 58
72 F=(TG-G)/TG : E=Y(J)-F : RETURN
```

NONLINEAR REGRESSION ANALYSIS

```
DATA FILE NAME                  AGAB201/DAT
MAX. NO. OF ITERATIONS           15
INITIAL VALUE OF LAMBDA          1
NO. OF PARAMETERS                3
```

STARTING PARAMETERS

```
B( 1 )          1E+06
B( 2 )          10000
B( 3 )          .4
```

OBSERVATIONS Y,X(1-5)

```
( 1 )   .336      1          0        0        0        0
( 2 )   .271      .5         0        0        0        0
( 3 )   .247      .25        0        0        0        0
( 4 )   .232      .2         0        0        0        0
( 5 )   .231      .166667    0        0        0        0
( 6 )   .21       .125       0        0        0        0
( 7 )   .196      .1         0        0        0        0
( 8 )   .147      .05        0        0        0        0
( 9 )   .122      .0333333   0        0        0        0
( 10 )  .108      .025       0        0        0        0
( 11 )  .093      .02        0        0        0        0
```

```
SSE= .259762          ITERATION NO.= 1          LAMBDA= 1
R-SQUARE =-3.5855
PARAMETERS
3.42768E+06  37845.6  .589247

SSE= .0276888         ITERATION NO.= 2          LAMBDA= .1
R-SQUARE = .511219
PARAMETERS
4.98381E+06  62957.1  .649253

SSE= 6.29858E-04      ITERATION NO.= 3          LAMBDA= .01
R-SQUARE = .988882
PARAMETERS
4.47224E+06  62553.9  .670344

SSE= 3.31247E-04      ITERATION NO.= 4          LAMBDA= 1E-03
R-SQUARE = .994153
PARAMETERS
4.42239E+06  60377.4  .674733

SSE= 3.26629E-04      ITERATION NO.= 5          LAMBDA= 1E-04
R-SQUARE = .994234
PARAMETERS
```

EXAMPLE (3) - 2 SPECIES ANTIBODY, 2 BINDING CONSTANTS - (3 PARAMETERS)

```
4.43023E+06   60681.1   .6743
PARAMETERS
4.4299E+06   60673.7   .674314
PARAMETERS
4.42752E+06   60619.5   .674418
PARAMETERS
4.42257E+06   60491.7   .674649

SSE= 3.25614E-04          ITERATION NO.= 6          LAMBDA= .01
R-SQUARE = .994234
PARAMETERS
4.40268E+06   59897.8   .675587
PARAMETERS
4.41817E+06   60265.9   .674898
PARAMETERS
4.42229E+06   60437.9   .674663

SSE= 3.26613E-04          ITERATION NO.= 7          LAMBDA= .1
R-SQUARE = .994234
PARAMETERS
4.42156E+06   60452.2   .67469
PARAMETERS
4.42223E+06   60447.9   .674671

CONVERGENCE

SSE= 3.26613E-04          ITERATION NO.= 8          LAMBDA= 0
R-SQUARE = .994234


VAR= 4.08266E-05          SD= 6.38958E-03           SSE= 3.26613E-04
```

FINAL PARAMETERS	STD. ERROR OF PARAM.	COEFF. OF VAR.
4.42229E+06	550569	.124499
60437.9	12239.4	.202512
.674663	.0223403	.0331133

VARIANCE-COVARIANCE MATRIX

```
3.03127E+11   5.1076E+09 -11292.8
5.1076E+09   1.49804E+08 -248.384
-11292.8 -248.384   4.9909E-04
```

PRINTOUT OF ESTIMATED Y FOR EACH X

X	Y	ESTIMATED Y -(2 * S.E.)	S.E. EST. Y +(2 * S.E.)	RESIDUAL
1	.336	.329505 .317819	5.84322E-03 .341192	6.49476E-03
.5	.271	.284716 .278255	3.2307E-03 .291177	-.013716
.25	.247	.245774 .239724	3.0248E-03 .251823	1.22634E-03
.2	.232	.233536 .227673	2.93133E-03 .239399	-1.53597E-03
.166667	.231	.223414 .217815	2.7993E-03 .229012	7.58627E-03
.125	.21	.207019 .201824	2.59741E-03 .212214	2.98081E-03
.1	.196	.193848 .188853	2.49761E-03 .198843	2.15201E-03

EXAMPLE (3) - 2 SPECIES ANTIBODY, 2 BINDING CONSTANTS - (3 PARAMETERS)

.05	.147	.150552 .145066	2.74308E-03 .156038	-3.55201E-03
.0333333	.122	.12463 .118288	3.17106E-03 .130972	-2.6303E-03
.025	.108	.106789 .100232	3.27849E-03 .113346	1.21116E-03
.02	.093	.0936081 .0069725	3.31783E-03 .100244	-6.08131E-04

TABLE OF RESIDUALS

(1)	6.49476E-03
(2)	-.013716
(3)	1.22634E-03
(4)	-1.53597E-03
(5)	7.58627E-03
(6)	2.98081E-03
(7)	2.15201E-03
(8)	-3.55201E-03
(9)	-2.6303E-03
(10)	1.21116E-03
(11)	-6.08131E-04

CORRELATION MATRIX

.999996	.459018	.841831
.459018	1	.822007
.841831	.822007	1

R= .390168 (1-R)= .609832

DERIVATION OF 1:1 ANTIGEN-ANTIBODY BINDING FUNCTION

Definitions:

G=free antigen conc. B=free antibody conc.
TG=total antigen conc. TB=total antibody conc.
BG=bound antigen-antibody complex K=binding constant

Equilibrium expression:

K=BG/(B*G) or BG=K*B*G (1)

Mass balance equations:

TB=B+BG (2) and TG=G+BG (3)

Using equations (1) and (3) we get:

TG=G+K*B*G or TG=G*(1+K*B) or G=TG/(1+K*B) (4)

Using equations (2) and (3) we get:

TB=B+K*B*G and using (4): TB=B+(K*B*TG/(1+K*B)) (5)

Using equations (2) and (5) we get:

BG=TB-B or BG=B+(K*B*TG/(1+K*B))-B or BG=K*B*TG/(1+K*B)

Dividing both sides by TG we get:

(BG/TG)=K*B/(1+K*B) (6)

(BG/TG) is the fraction antigen bound or our measured Y value. If we let K*B equal X, we see that the 1:1 binding equation reduces to the hyperbolic form Y=X/(1+X), where Y=fraction antigen bound and X=the binding constant times the free antibody concentration.

In order to solve equation (6) for fraction antigen bound, we must first know the free antibody concentration (B). We can use equation (5) which contains only known quantities and B, and reduce it to a second order polynomial in B.

Multiplying each term of equation (5) by (1+K*B) we get:

TB*(1+K*B)=B*(1+K*B)+K*B*TG or TB+K*B*TB=B+K*B*B+K*B*TG

Rearranging and collecting terms we get:

K*(B*B)+B+K*B*TG-K*B*TB-TB=0 or K*(B*B)+(1+K*(TG-TB))*B-TB=0

This is a quadratic equation in B where the terms are;

a=K b=1+K*(TG-TB)) and c=-TB

It is now possible to use the quadratic equation ((-b+SQR(b*b-4*a*c))/2*a) to solve for B using these terms. These are the same terms used in function AGAB/FUN in Appendix A and in examples 1 and 2 Appendix B. Once B is known then equation (6) can be used to calculate the fraction antigen bound and the error term to complete the function subroutine. This same procedure can be used to solve other equilibrium models provided a means for finding the root of the derived polynomial is available.

APPENDIX C

MODEL FOR ANTIGEN-ANTIBODY BINDING ASSUMING TWO DISTINCT SPECIES OF ANTIBODY WITH DIFFERENT BINDING CONSTANTS FOR ANTIGEN

Definitions:

G=free antigen conc.
TB1=total conc. antibody 1
TB2=total conc. antibody 2
TB=total antibody conc.
TG=total antigen conc.
GB1=complex with B1
B1=free antibody 1 conc.
B2=free antibody 2 conc.
R=fraction of TB as TB1
K1=binding constant 1
K2=binding constant 2
GB2=complex with B2

Equilibrium expressions:

GB1=K1*G*B1 and GB2=K2*G*B2

Mass balance equations:

TB1=B1+GB1 and TB2=B2+GB2 and TG=G+GB1+GB2

Partition functions:

TB=TB1+TB2 and TB1=R*TB and TB2=(1-R)*TB

Using the equilibrium and mass balance equations we get:

B1=TB1/(1+K1*G) or B1=R*TB/(1+K1*G) (1)

B2=TB2/(1+K2*G) or B2=(1-R)*TB/(1+K2*G) (2)

TG=G+K1*G*B1+K2*G*B2 or TG=G*(1+K1*B1+K2*B2) (3)

If we were to solve these equations in terms of G we would obtain a third order polynomial or cubic equation in G. Since there is no simple technique for obtaining the root of this equation, there is no point in solving in terms of G. Instead we use a numerical technique for solution which depends on the equations being positive definite functions.

Equations (1)-(3) above are structured so that given G we can solve for TG. TG is known however and is the key to knowing when we have selected the correct G. First we set a lower and an upper limit to G. We set the lower limit to 0 and the upper limit to TG. We then guess that the value for G is half way between 0 and TG. This first value of G is used to calculate a value for TG and compare it to the real value. If the calculated value is too high then B was too high and we make the first guess on G the new upper limit. If it is too low we make the first guess on G the new lower limit. We make a new guess half way between the new upper and lower limits and try again until the value for G converges to the correct value. This usually happens in 7-10 passes. Once we know the value for G we can calculate the fraction bound (F=(TG-G)/TG).

The following BASIC code performs the numerical solution of a function using this binary search method:

```
50 REM ** FUNCTION FOR 2 DIFFERENT SPECIES OF ANTIBODY WITH
DIFFERENT K'S --- P1=K1 AND P2=K2 **
52 P1=B(1) : P2=B(2) : TB=X(J,1)*4.5E-6 : TG=X(J,1)*6.9E-6
54 BL=0 : LG=0 : TL=TG
56 R=SIN(B(3))*SIN(B(3))
58 G=.5*(BL+TL)
60 B1=R*TB/(1+P1*G)
62 B2=(1-R)*TB/(1+P2*G)
64 TC=G*(1+P1*B1+P2*B2)
66 IF TC>TG THEN TL=G ELSE BL=G
68 IF ABS((LG-G)/G)<1E-6 THEN 72
70 LG=G : GOTO 58
72 F=(TG-G)/TG : E=Y(J)-F : RETURN
```

The variables are defined as follows:

P1=K1, P2=K2, R= fraction total antibody as species B1.
B(1), B(2) and B(3) are the parameter variables B's manipulated by the main program representing K1, K2, and R respectively.
TB and TG are the total antibody and total antigen concentrations.
X(J,1) is the X data variable, J is the number of the data point and 1 means the first independent variable of a possible 5.
G is the free antigen concentration.
BL is the bottom limit for G, TL is the top limit for G and LG is the last value for G.
B1 and B2 are the free concentrations of antibodies 1 and antibody 2 respectively.
TC is the calculated concentration for TG, TG(calc).
Y(J) is the Y value for the Jth data point.
Line 58 provides a new guess for G half way between the upper and the lower limits.
Lines 60 to 64 calculate a value for TG(calc) using the G from line 58.
Line 66 compares the TG(calc) with TG and sets the new upper or lower limit depending on the outcome.
Line 68 tests for convergence and either branches out of the loop or causes another pass.
Line 70 resets the last G value to the current value.

VARIABLE CROSS REFERENCE INDEX FOR "NONLIN/PRT"

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 83-5	2. GOVT ACCESSION NO. AD-A126840	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NON-LINEAR REGRESSION ANALYSIS: A GENERAL PROGRAM FOR DATA MODELING USING PERSONAL MICROCOMPUTERS		5. TYPE OF REPORT & PERIOD COVERED FINAL 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) LCDR D. NELSON - NHRC Dr. Louis D. Homer - NMRI, Bethesda, MD		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Health Research Center P.O. Box 85122 San Diego, CA 92138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 3M162770A871.AB.306
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Medical Research & Development Command National Naval Medical Center Bethesda, MD 20814		12. REPORT DATE Feb. 1983 13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Commander, Naval Medical Command Department of the Navy Washington, DC 20372		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Non-Linear Regression Analysis
Antigen-Antibody Complexes
Microcomputers
Data Modeling
Basic Program

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents a general non-linear regression program for fitting data to non-linear models. The program is based on an algorithm by Marquardt which uses a least squares criterion to calculate successive improvements to an initial set of parameter estimates. The program is written in the BASIC language common to most microcomputers, because it is easy to use and to transport between machines from different manufacturers. The majority of inexpensive microcomputers do not offer matrix operations as part of their BASIC interpreter. The program presented here, therefore, supplies subroutines in BASIC

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Item 20 (cont'd)

for the zeroing, transposing and inverting of the required matrices, to make it compatible with most microcomputers available today. The report gives examples and program output based on a demonstration data set involving antigen-antibody complexation in solution. Two derivations of function subroutines are given to assist the user in designing his own function subroutines. A complete listing of the necessary programs is given along with a section on program cautions.